УДК 004.896

С.С. Синельников
Государственный университет информатики и искусственного интеллекта,
г. Донецк, Украина
ssergs82@mail.ru

Теоретические основы выбора оптимального метода поиска в несбалансированном бинарном дереве

Проведен теоретический анализ методов поиска для бинарных деревьев; предложена вероятностная модель движения по бинарному дереву, позволяющая определить лучший метод поиска; решена задача выбора оптимального метода поиска в бинарном дереве с учетом статистики обращений к его элементам.

Введение

С ростом требований к скорости выполнения задачи поиска и сортировки данных в интеллектуальных системах их разработчики все чаще используют для решения этой задачи различные древовидные структуры. В связи с чем возникают проблемы, связанные с понижением вычислительной сложности алгоритма поиска данных, выбором оптимального метода поиска и балансировкой деревьев [1-4]. Данные проблемы достаточно важны, а их решения на данный момент не имеют серьезной теоретической основы и носят частный характер.

Второй важной задачей в интеллектуальных системах есть задача выбора оптимального метода поиска с учетом статистики обращений к каждому элементу бинарного дерева, решение которой позволяет повысить интеллектуальность системы, применяющей его, и увеличить скорость выполнения задачи поиска.

В данной работе предлагается теоретическая основа (методы теории вероятностей [5], [6]) для описания модели движения по бинарному дереву, которая позволяет решить задачу выбора оптимального метода поиска данных, а также задачу выбора оптимального метода поиска данных с учетом статистики обращений к каждому элементу бинарного дерева. Будет показана возможность применения данной модели к бинарному методу поиска, используемого в массивах.

Выбор оптимального метода поиска в несбалансированном бинарном дереве на основе анализа его структуры

Построение сбалансированного бинарного дерева – одна из важнейших задач, решение которой позволяет применять методы поиска с большей эффективностью. Но ее решение связано с трудностями, которые приводят к большому количеству вычислительных затрат. Поэтому возникает задача поиска данных в несбалансированном бинарном дереве с наименьшими затратами, в частности с минимальным количеством сравнений.

Классическая реализация алгоритма поиска в дереве выглядит следующим образом (рис. 1).

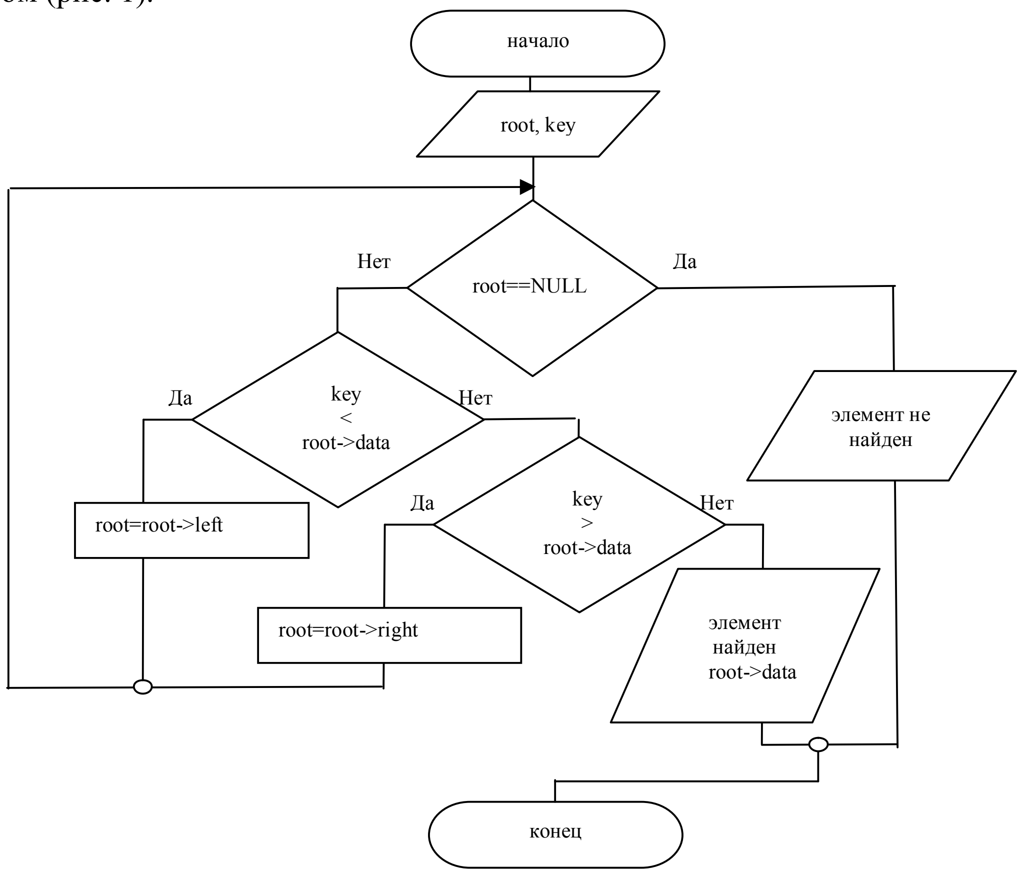


Рисунок 1 – Метод поиска в бинарном дереве search_in_tree1

Проанализируем работу данного метода на различных топологиях несбалансированного бинарного дерева. Для случая, продемонстрированного на рис. 2а), метод search_in_tree1 ведет себя как линейный поиск и тратит минимум сравнений, в среднем N/2 сравнений для поиска. Это происходит благодаря тому, что работа алгоритма начинается с условия проверки «<», что и определило успех.

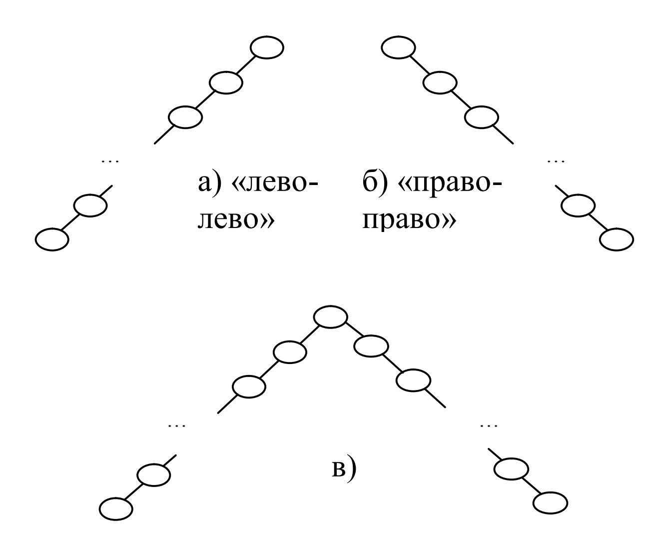


Рисунок 2 – Топологии несбалансированного бинарного дерева

Заметим, что для топологии а) проверка условия «>» вообще никогда не производится при поиске заведомо имеющегося элемента.

Рассмотрим вариант, представленный на рис. 2б). Для него метод search_in_tree1 ведет себя наихудшим образом, выполняя при этом максимум сравнений – в среднем их количество равно N. Данная ситуация происходит из-за того, что проверка «<» ни разу не срабатывает, что приводит к переходу на условие «>».

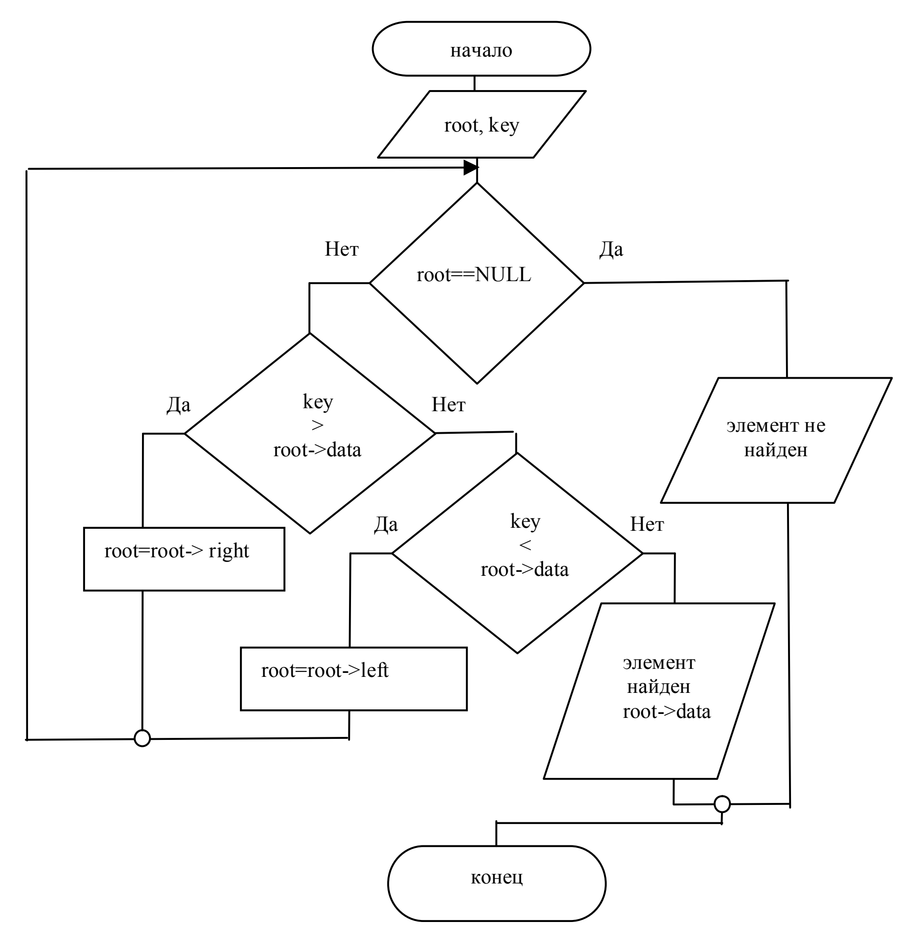


Рисунок 3 – Метод поиска в бинарном дереве search_in_tree2

Очевидно, что для случая б) логично выполнять сначала проверку на «>», а затем на «<». С таким подходом получим метод search_in_tree2, который для топологии б) даст в среднем N/2 сравнений. Для топологии а) метод search_in_tree2 будет выполнять максимум сравнений.

Таким образом, для каждой топологии лучше выбирать определенный метод. Заметим, что для достаточно сбалансированного дерева любой из этих методов будет иметь равные показатели. Худшим вариантом для методов search_in_tree1 и search_in_tree2 является топология вида рис. 2в). Для нее методы выполняют в среднем порядка ¾N сравнений. Таким образом, приходим к выводу, что для несбалансированных деревьев методы search_in_tree1 и search_in_tree2 не подходят и требуется новый алгоритм, не имеющий этих недостатков.

Для реализации такого метода понадобится циклический подход к решению задачи – поиск в одном направлении (только по левому поддереву или только по правому). Как только поиск по одному направлению невозможен, переходим на другое направление. Реализация метода представлена на рис. 4.

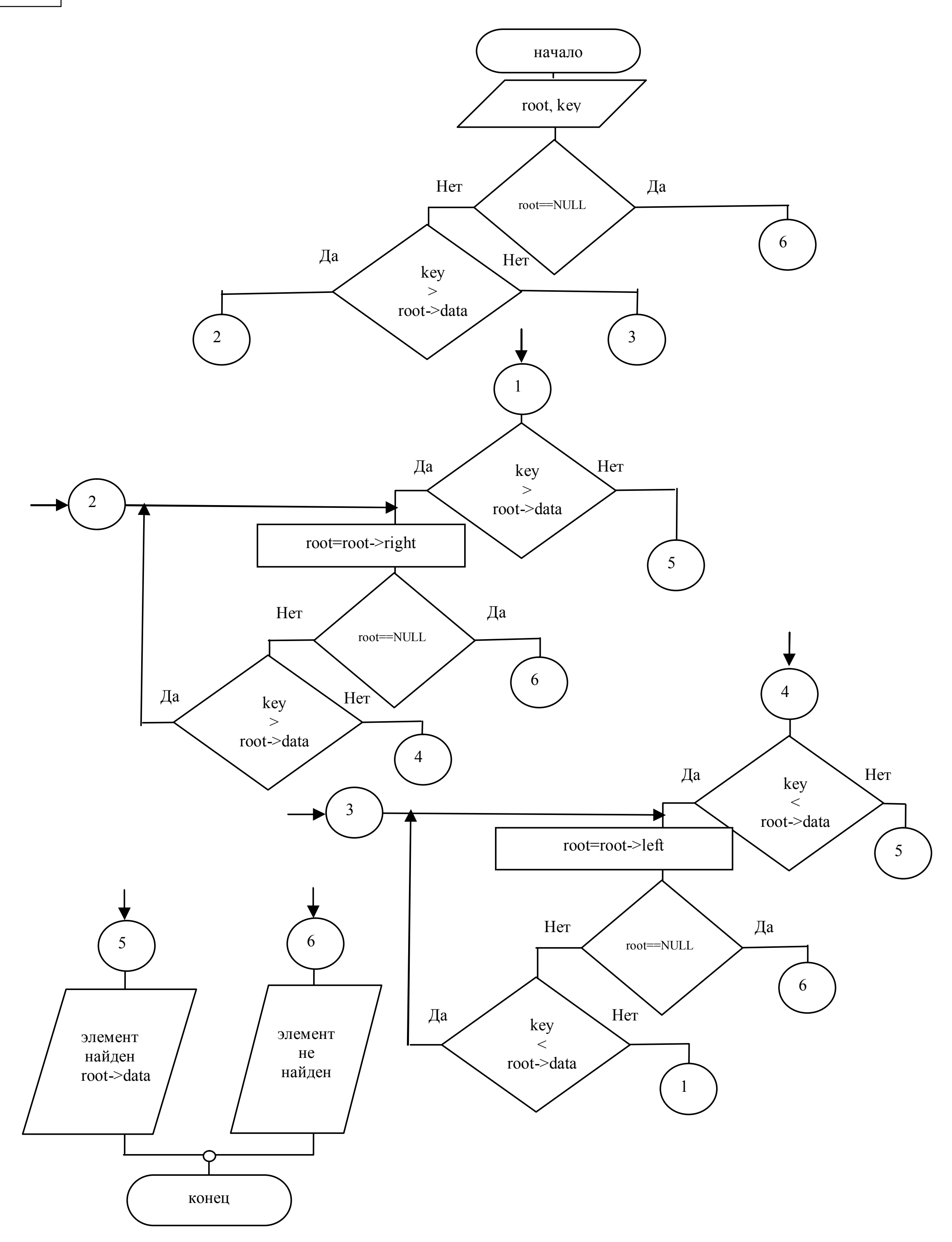


Рисунок 4 – Блок-схема алгоритма search_in_tree_new

Метод search_in_tree_new выполняет стабильно в среднем порядка N/2 сравнений для каждой топологии, представленных на рис. 2. Недостатком выступает то, что для деревьев, в которых «правые» элементы имеют одного «левого» сына или «левые» элементы имеют одного «правого» сына, скорость поиска данного метода значительно увеличивается.

Особенно наглядна эта ситуация на топологиях, представленных на рис. 5. В этих случаях происходит чередование движения – «влево-вправо», что приводит к среднему количеству сравнений для варианта б) и в) – порядка N.

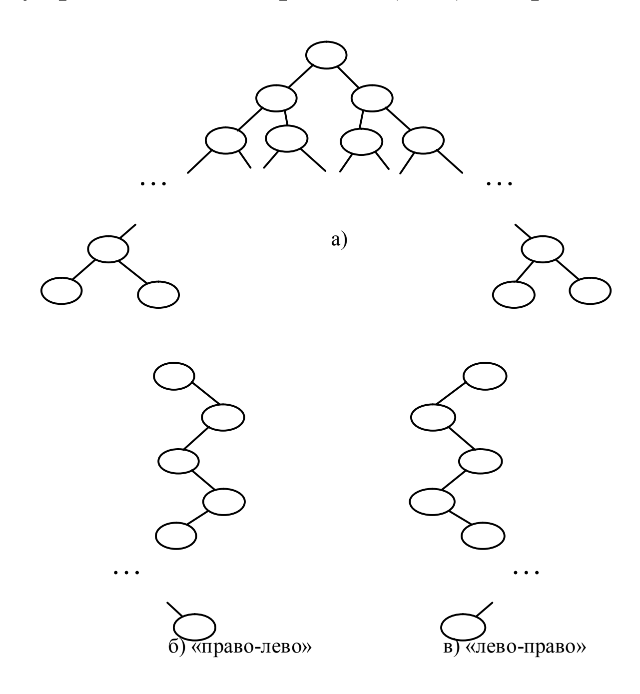


Рисунок 5 – Топологии сбалансированного а), несбалансированного б) и в) бинарного дерева

Для варианта а) – сбалансированного дерева – порядок производимых сравнений тот же, как и для методов search_in_tree1 и search_in_tree2, и соответствует порядку logN.

Таким образом, для различных топологий имеем методы, работающие с разной эффективностью, и задача сводится к объединению этих методов для получения наилучшего результата. Очевидно, что такой метод должен учитывать статистические данные о дереве, каких топологий больше, и в соответствии с этой информацией выбирать наилучший метод.

Для деревьев с доминированием топологии «право-лево» или «право-лево» лучше применять методы search_in_tree1 и search_in_tree2, а при доминировании топологий «лево-лево» или «право-право» – метод search_in_tree_new. Такой подход при решении задачи сортировки позволит уменьшить количество сравнений без применения механизма балансировки деревьев. Недостаток метода – перед применением необходимо произвести просчет количества левых и правых сыновей или различных топологий.

Вероятностная модель обхода бинарного несбалансированного дерева

В рассуждениях, проведенных выше, не учитывалась особенность расположения элемента – чем выше он расположен и чем больше у него сыновей, тем чаще при поиске через него проходит путь к искомым данным. Рассмотрим вероятностную модель обхода несбалансированного бинарного дерева при поиске элемента.

Будем считать, что все элементы бинарного дерева различны (копий нет).

Возьмем один элемент и рассмотрим его вероятностную модель переходов (рис. 6). На рис. 6 представлены: pv – вероятность попадания на элемент, p – вероятность того, что это искомый элемент, x – вероятность перехода по левой ветке, y – вероятность перехода по правой ветке. Тогда

$$p = \frac{1}{N} \cdot pv,$$

где N – количество элементов поддерева

$$\begin{cases} x = a \cdot pv - p \cdot \dfrac{x}{x+y} \\ y = b \cdot pv - p \cdot \dfrac{y}{x+y} \end{cases},$$

где

$$a = \frac{Nl}{Nl + Np},$$

$$b = \frac{Np}{Nl + Np},$$

Nl – количество элементов левого поддерева, Np – количество элементов правого поддерева (Nl + Np = N).

С учетом того, что p + x + y = pv, получаем:

$$\begin{cases} x = a \cdot (pv - p) \\ y = b \cdot (pv - p). \end{cases}$$

Рассмотрим общий случай. Пусть $p_{i,j}$ – вероятность того, что j-й элемент i-го уровня является искомым, а $pv_{i,j}$ – вероятность перехода на j-й элемент (i + 1)-го уровня. Тогда

$$p_{i,j} = \frac{1}{N} \cdot pv_{i-1,j},$$

где N – количество элементов поддерева с вершиной у j-го элемента i-го уровня

$$pv_{i,j} = \begin{cases} a \cdot (pv_{i-1,(j+1)/2} - p_{i,(j+1)/2}), \text{ если } j - \text{нечетное} \\ b \cdot (pv_{i-1,(j+1)/2} - p_{i,(j+1)/2}), \text{ если } j - \text{четное} \end{cases}.$$

Для каждого j-го элемента i-го уровня справедливо:

$$p_{i,j} + pv_{i,j*2-1} + pv_{i,j*2} = pv_{i-1,(j+1)/2},$$

что обеспечивает охват всех возможных шагов. Сумма всех вероятностей $p_{i,j}$ равна 1, что позволяет всегда найти искомый элемент.

На рис. 7 и 8 представлен наглядный пример, позволяющий рассчитать вероятность перехода по каждой «ветке». Данные вероятностей перехода позволяют оценить, какой из методов наиболее эффективен для поиска. Например, количество левых ветвей равно 5, а правых – 6, но использовать метод поиска search_in_tree2, начинающийся с проверки на «>», не выгодно.

Сравним, что больше: вероятность движения влево или вправо. Сумма вероятностей перехода влево составляет 1,75, а вправо – 0,75. Количество сравнений для поиска всех элементов, произведенных методом search_in_tree1, начинающимся с проверки на «<», составляет 63, а методом search_in_tree2, начинающимся с проверки на «>», – 75.

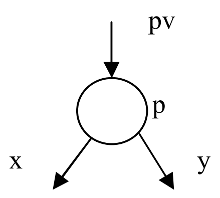


Рисунок 6 – Вероятностная модель переходов для одного элемента

$pv_{0,1} = 1$
$p_{1,1} = 1/12$
1
$pv_{1,1} = 9/11*(pv_{0,1} - p_{1,1})$
$pv_{1,2} = 2/11*(pv_{0,1} - p_{1,1})$
$p_{2,1} = 1/9*pv_{1,1}$
2
$p_{2,2} = 1/2*pv_{1,2}$
3
$pv_{2,1} = 7/8*(pv_{1,1} - p_{2,1})$
$pv_{2,2} = 1/8*(pv_{1,1} - p_{2,1})$
$p_{3,2} = 1/1*pv_{2,2}$
$pv_{2,4} = 1/1*(pv_{1,2} - p_{2,2})$
$p_{3,1} = 1/7*pv_{2,1}$
4
5
NULL
$p_{3,4} = 1/1* pv_{2,4}$
NULL NULL
6
$pv_{3,1} = 3/6*(pv_{2,1} - p_{31})$
$pv_{3,2} = 3/6*(pv_{2,1} - p_{3,1})$
NULL NULL
$p_{4,1} = 1/3*pv_{3,1}$
7
$p_{4,2} = 1/3*pv_{3,2}$
$pv_{4,1} = 1/2*(pv_{3,1} - p_{4,1})$
$pv_{4,2} = 1/2*(pv_{3,1} - p_{4,1})$
8
$p_{5,1} = 1/1* pv_{4,1}$
$p_{5,2} = 1/1* pv_{4,2}$
$pv_{4,4} = 1/2*(pv_{3,2} - p_{4,2})$
9
10
$pv_{4,3} = 1/2*(pv_{3,2} - p_{4,2})$
$p_{5,3} = 1/1* pv_{4,3}$
$p_{5,4} = 1/1* pv_{4,4}$
NULL NULL
11
12
NULL NULL
NULL NULL
NULL NULL

Рисунок 7 – Вероятностная модель обхода несбалансированного бинарного дерева

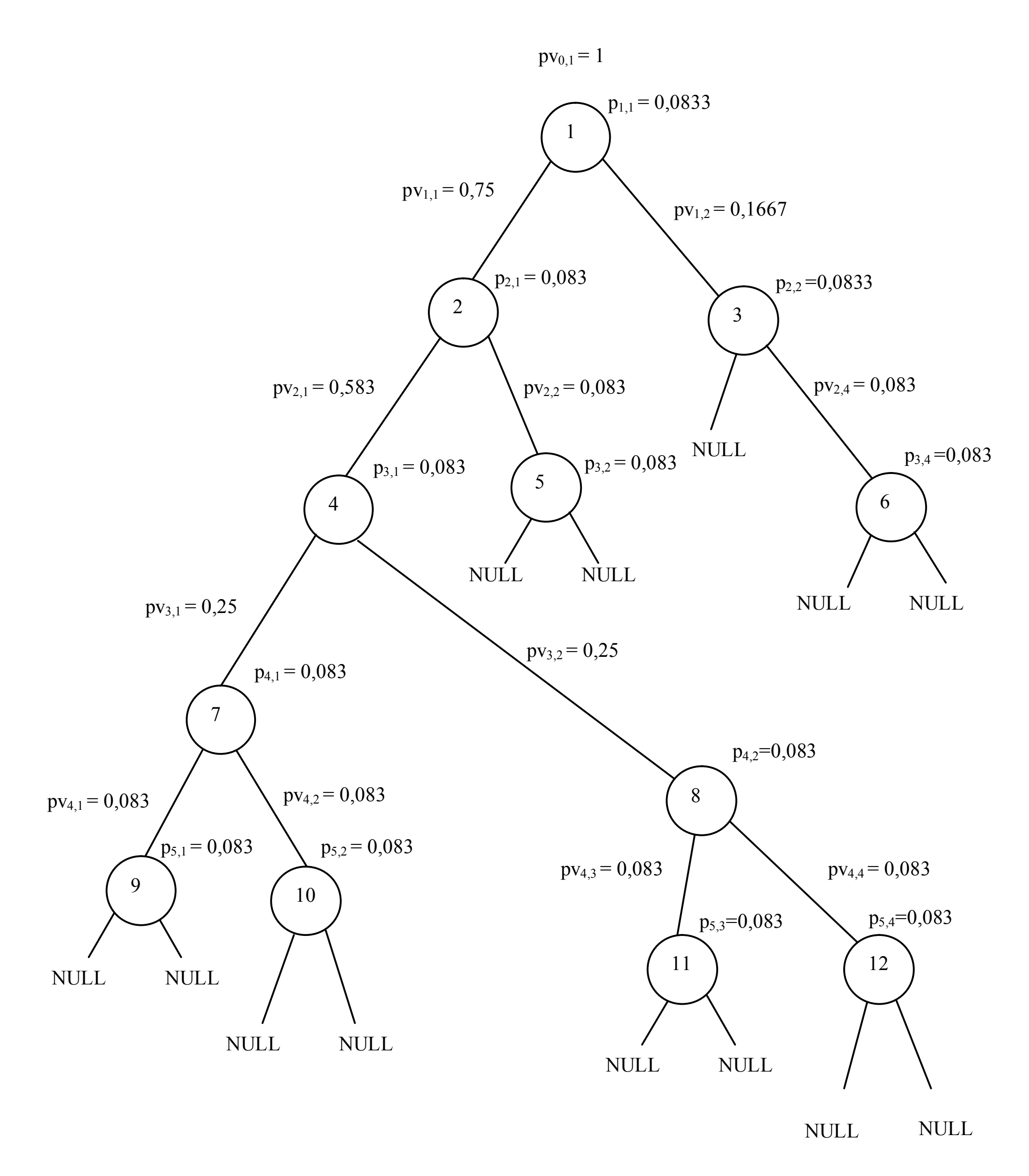


Рисунок 8 – Вероятностная модель обхода несбалансированного бинарного дерева с расчетом вероятностей

Итак, вероятностная модель обхода несбалансированного бинарного дерева обеспечила правильный ответ с учетом расположения элементов, количества их сыновей и элементов в поддереве.

Таким образом, данная модель позволяет выбрать, с какого сравнения начать: «>» или с «<», что позволяет определить, какой алгоритм будет эффективнее: search_in_tree1 или search_in_tree2, search_in_tree_new, начинающийся с проверки «>», или «<».

Полученная модель может быть использована не только для выбора оптимального метода поиска, но и для задачи сортировки с применением бинарного дерева, а также служить основой для решения задачи поиска с использованием статистических особенностей – в частности количества обращения к каждому элементу дерева.

Решение задачи выбора оптимального метода поиска данных с учетом статистики обращений к каждому элементу

Рассмотрим задачу выбора оптимального метода для поиска данных с заранее известными вероятностями обращения к каждому элементу и покажем, что данная задача может быть решена с помощью вероятностной модели движения по несбалансированному бинарному дереву.

Действительно, если рассматривать количество обращений к определенному элементу не как один элемент, а как количество элементов, то получаем задачу выбора оптимального метода поиска, которая решается с помощью вероятностной модели движения по несбалансированному бинарному дереву.

Пусть $p_{i,j}$ – вероятность того, что j-й элемент i-го уровня является искомым, а $pv_{i,j}$ – вероятность перехода на j-й элемент (i+1)-го уровня. Тогда

$$p_{i,j} = \frac{1}{N} \cdot pv_{i-1,j}, \quad \text{если} \quad N > 0,$$

где N – количество элементов поддерева с вершиной у j-го элемента i-го уровня

$$pv_{i,j} = \begin{cases} a \cdot (pv_{i-1,(j+1)/2} - p_{i,(j+1)/2}), \\ \text{если} \quad j - \text{не четное} \quad \text{и} \quad Nl <> 0 \\ \\ b \cdot (pv_{i-1,(j+1)/2} - p_{i,(j+1)/2}), \\ \text{если} \quad j - \text{четное} \quad Np <> 0 \\ \\ 0, \quad \text{иначе.} \end{cases}$$

Для каждого j-го элемента i-го уровня справедливо:

$$p_{i,j} + pv_{i,j*2-1} + pv_{i,j*2} = pv_{i-1,(j+1)/2},$$

что обеспечивает охват всех возможных шагов. Сумма всех вероятностей $p_{i,j}$ равна 1, что позволяет всегда найти искомый элемент.

Рассмотрим наглядный пример (рис. 9), позволяющий оценить вероятность перехода по каждой «ветке» с учетом количества обращений к каждому элементу (табл. 1).

Таблица 1 – Количество обращений к каждому элементу

№ элемента	1	2	3	4	5	6	7	8	9	10	11	12
Количество обращений	0	4	1	0	2	3	1	5	1	1	0	1

Сравним, что больше вероятность движения влево или вправо. Сумма вероятностей перехода влево составляет 0,7894737 + 0,4736 + 0,1578 + 0,05263 = 1,4736, а вправо = 0,2105263 + 0,1052 + 0,1578 + 0,3157 + 0,05263 + 0,05263 = 0,89473.

Количество сравнений для поиска всех элементов с учетом статистики (табл. 1), произведенных методом search_in_tree1, начинающимся с проверки на «<», составляет 100, а методом search_in_tree2, начинающимся с проверки на «>», – 111.

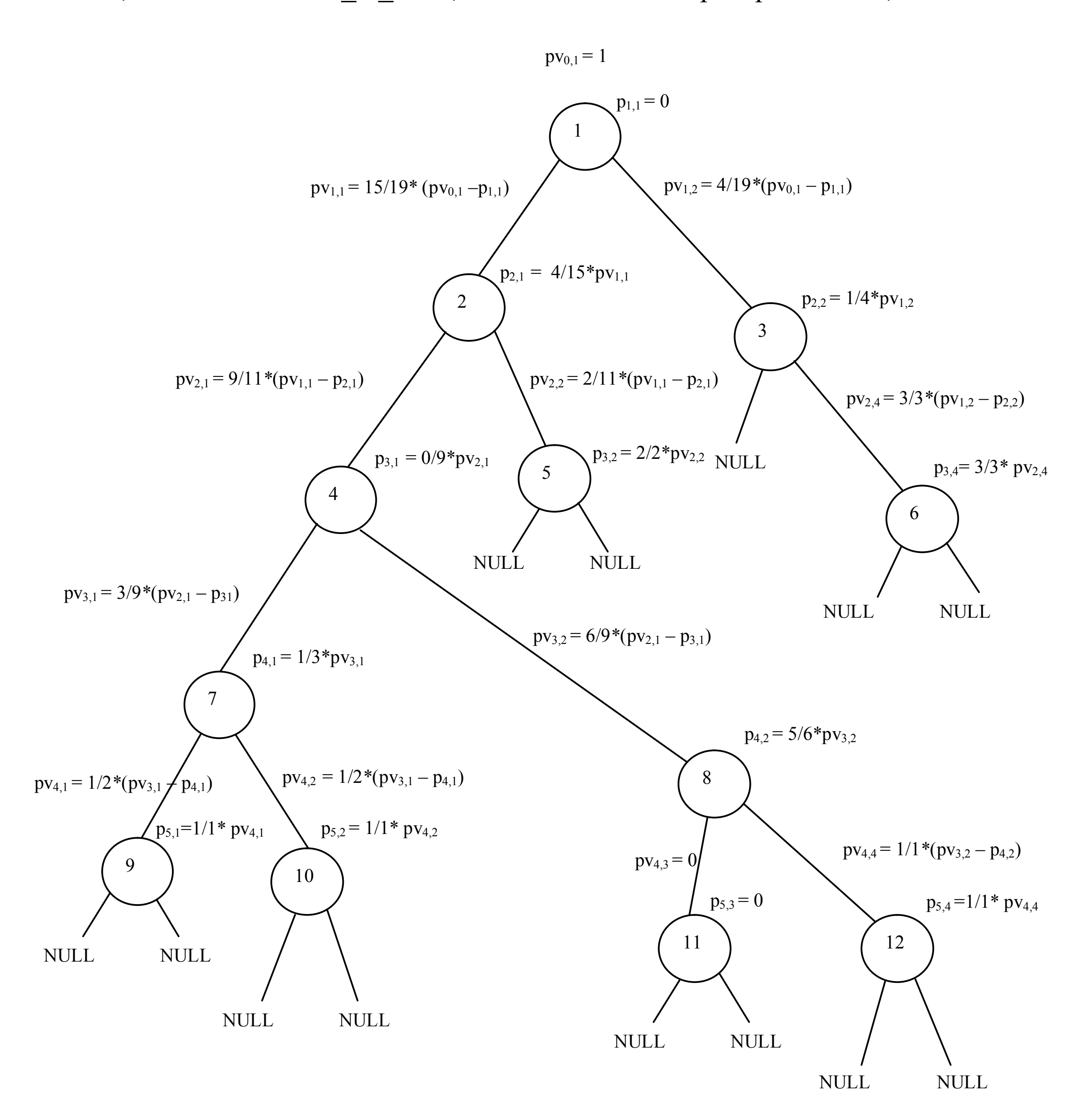


Рисунок 9 – Вероятностная модель обхода несбалансированного бинарного дерева с учетом количества обращений к каждому элементу

Таким образом, вероятностная модель обхода несбалансированного бинарного дерева обеспечила правильный ответ не только с учетом расположения элементов, количества их сыновей и элементов в поддереве, а и дополнительных статистических особенностей, в частности количества обращений к каждому элементу.

Полученная модель может быть использована для выбора оптимального метода поиска, для задачи сортировки с применением бинарного дерева, а также служить основой для решения задачи выбора оптимального метода поиска с использованием статистических особенностей из класса бинарных алгоритмов, которые по принципу работы сходны с алгоритмами поиска в бинарных деревьях.

Выводы

Разработаны методы поиска для бинарных деревьев с минимальным количеством сравнений. Предложена теоретическая основа – методы теории вероятностей – для описания модели обхода несбалансированного бинарного дерева, которая позволяет решить задачу выбора оптимального метода поиска данных, а также задачу выбора оптимального метода поиска данных с учетом статистики обращений к каждому элементу бинарного дерева. Показана возможность применения данной модели к бинарному методу поиска, используемого в массивах.

В перспективе планируется анализ использования полученной модели для процесса балансировки деревьев, анализа графов.

Полученные результаты позволяют увеличить скорость выполнения задачи поиска, а также повысить интеллектуальность системы, применяющей новые методы поиска и анализа данных.

Литература

1. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. – 2-е издание. – Вильямс, 2005. – 1296 с.
2. Кнут Д. Искусство программирования. – 3-е издание. – Вильямс, 2005. – 720 с.
3. Седжвик Р. Фундаментальные алгоритмы на С++: Ч. 1 – 4: Анализ, структуры данных, сортировка, поиск: Пер. с англ. – Diasoft. – 2001. –687 с.
4. Бьерн Страуструп. Язык программирования С++. – М.: Бином, 2005.
5. Колмогоров А.Н. Основные понятия теории вероятностей. – М.: Наука, 1974. – 120 с.
6. Гмурман В.Е. Теория вероятности и математическая статистика: Учебное пособие. – М.: Высшая школа, 2005. – 479 с.

С.С. Синельников

Теоретичні основи вибору оптимального методу пошуку в незбалансованому бінарному дереві

У статті проведений теоретичний аналіз методів пошуку для бінарних дерев; запропонована ймовірнісна модель руху по бінарному дереву, яка дозволяє визначити найкращий метод пошуку; розв’язана задача вибору оптимального методу пошуку в бінарному дереві з урахуванням статистики звернень до елементів.

Статья поступила в редакцию 18.06.2008.